Homework 8 Selected Solutions
APPM/MATH 4650 Fall '20 Numerical Analysis

Due date: Saturday, November 7, before midnight, via Gradescope. **Instructor**: Prof. Becker
Theme: ODEs and IVPs, and Halloween

solutions version 10/30/2020

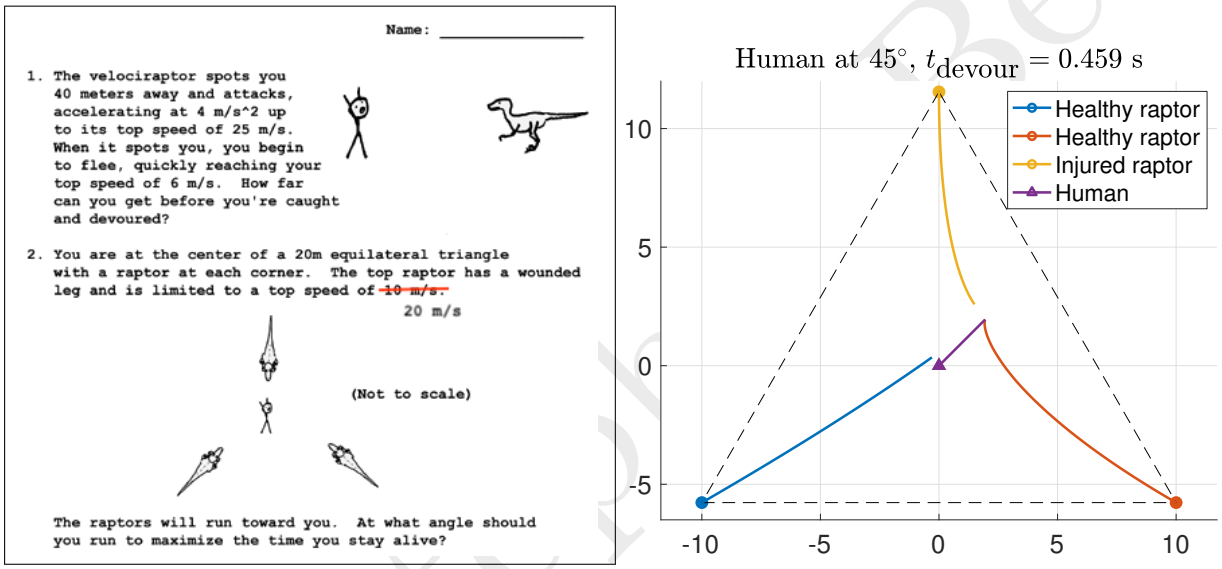

Figure 1: From the XKCD comic. **Left**: We'll solve problem 2. The full comic can be found at xkcd.com/135/. **Right**: An example of the output from solving the raptor ODE, when the human decides to run at a 45° angle above the horizontal. The healthy raptor in the bottom right corner reaches the human in 0.459 seconds. Your homework should look similar to this plot, but use an angle of 56°. All distances in meters.

Problem 1: Raptors Scientist/cartoonist Randall Munroe poses an interesting question about optimization in his XKCD comic. We can start to answer question 2 (see Figure 1) using Matlab or Python's builtin IVP solvers. First, formulate the problem as an ODE. Let the human position be $\mathbf{h}(t) \in \mathbb{R}^2$, and a raptor's position $\mathbf{r}(t) \in \mathbb{R}^2$. We assume a raptor is not good at predicting the human's future location, and at any time t, the raptor runs directly at the human's current position. If the raptor's speed is constant (for simplicity, we also assume instantaneous acceleration at the beginning) at, say, v_r, then we can model the raptor's motion as

$$\frac{d\mathbf{r}}{dt} = v_r \frac{\mathbf{h}(t) - \mathbf{r}(t)}{\|\mathbf{h}(t) - \mathbf{r}(t)\|_2} \tag{1}$$

where if $\mathbf{x} = (x, y)$ then we define $\|\mathbf{x}\|_2 = \sqrt{x^2 + y^2}$.

The three raptors each satisfy this ODE separately, i.e. their motions are not coupled. However, the x and y components of a raptor's motion are coupled, so we have 2-dimensional ODEs.

For simplicity, we assume the human runs in a constant direction and at a constant speed; thus $\mathbf{h}(t) = v_h t \frac{\mathbf{c}}{\|\mathbf{c}\|_2} + \mathbf{h}(0)$, where $\mathbf{c} \in \mathbb{R}^2$ is an initial direction and $\mathbf{h}(0)$ is the human's initial

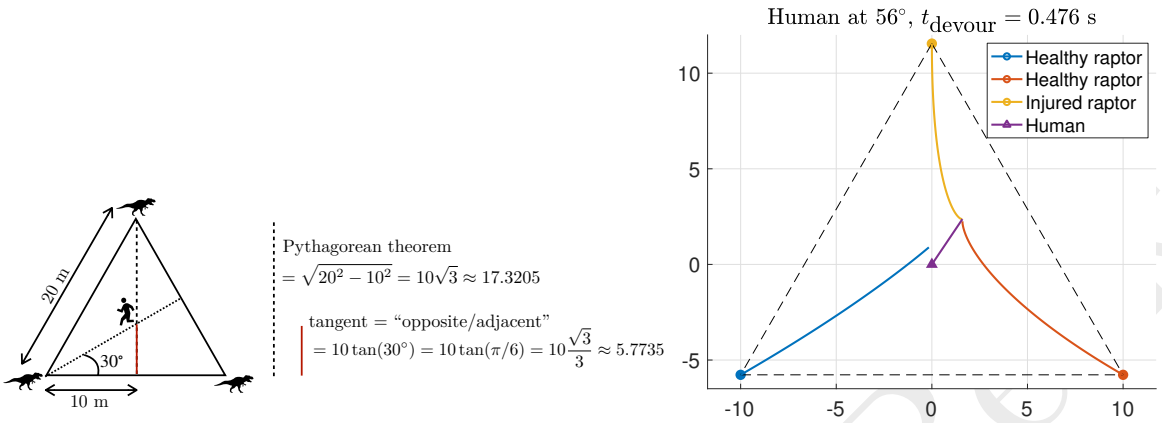

Figure 2: Left: Setting up the geometry of the problem. Right: solution.

position. Then, substitute $\mathbf{h}(t)$ into equation (1) to obtain $\frac{d\mathbf{r}}{dt} = F(t, \mathbf{r})$ for some function F that you must define.

We will use Munroe's setup, with a small modification (and assuming instantaneous acceleration). As in the comic, the human is at the center of a 20m equilateral triangle, and the human has a top speed of $v_h = 6$ m/s and the healthy raptors have a top speed of $v_r = 25$ m/s. However, if the injured raptor can only run at 10 m/s, then the best strategy for the human is to run directly at the injured raptor. To make the problem more interesting, assume the injured raptor can run at 20 m/s.

Your job is to setup the geometry and IVP, then using an existing IVP solver (such as `ode45` in Matlab or `scipy.integrate.solve_ivp` in Python), solve the IVP and determine how long the human will survive. We will define the human to be "caught" by the raptor when the raptor is within 0.01 m of the human. Specifically, assume the human starts running at a 56° angle above the horizontal (to the right, in the picture), and find which raptor reaches the human first, and how long it takes that raptor to reach the human (report your answer in seconds, with 3 decimal places). You should also include a plot, similar to the one in Figure 1 (right), and use interpolation to plot the lines (e.g., `deval` in Matlab, and set the `dense_output=True` flag for `solve_ivp` in Python). Please include your code.

Hints: it may be helpful to tighten the tolerance for the IVP solvers. Also, you do not know *a priori* the ending time of the IVP. One nice way to deal with this is to pick a conservative time (say, $t = 5$), and then define an "event", such that the IVP solver will stop when the event occurs. Both `ode45` and `scipy.integrate.solve_ivp` support "events"; see their documentation page, or ask in office hours. Note that you don't have to optimize the angle for this problem, though that is an interesting question and not much more difficult for us to answer numerically. Another interesting question is for which top speed of the wounded raptor does your optimal strategy become to run straight at the wounded raptor?

Solution:

The answer is that if you run at an angle of 56°, then the injured raptor is the first to reach you, and it takes it $\boxed{0.476 \text{ seconds}}$. The geometry and plot are in the figure. Complete code in both Matlab and Python is at the end of the homework.

Some comments:

a) When t is sufficiently large, the raptor has caught the human, and $\mathbf{r}(t) = \mathbf{h}(t)$, which means the right-hand-side of Eq. (1) is 0/0 and is undefined. So the ODE does not have a solution past this time. However, if the tolerance of your solver isn't large enough, the solver time points "skip" over this point, and then afterwards, where the solution is undefined, we can't control what is going to happen. Whatever the solver gives us here is meaningless. So we

need to make the solver tolerances sufficiently small, and use the "event" flags to also help. (Using just interpolation with the event flag will not help, since we'll be interpolating from data based on the meaningless solution).

b) For setting the tolerances, a good idea would be to run the solver for different tolerances, seeing what the time is, and making the tolerance stricter and stricter until you stop seeing much change in the solver time. However, by using the "event" flag, the tolerance setting is less important, since the solvers will do root-finding to find where the event flag first crosses zero, and so this will automatically increase the accuracy near the end, as needed. The tolerance is still slightly important here, since if the tolerance is way too loose, then the solver might not notice that the "events" function changed sign (since who knows what the sign of the rhs is after the person is caught).

c) For plotting, for full credit, students should use interpolation.

Problem 2: Consider the IVP

$$\begin{aligned} y' &= \frac{1+t}{1+y}, \quad t \in [1,2] \\ y(1) &= 2 \end{aligned} \tag{2}$$

a) Prove there exists a unique solution y to this IVP for the time interval $[1, 1\frac{2}{3}]$. If you prove there exists a unique solution for the time interval $[1, 2]$, you get a small amount of extra credit. *Hint*: you may want to use the theorem provided at the end of this homework.

Solution:

Our first attempt at a solution might be by Theorem 5.4 in Burden and Faires[1], so we need to check that $f(t,y) = \frac{1+t}{1+y}$ is (1) jointly continuous in (t,y) and (2) Lipschitz continuous in y, uniformly in t, all on the domain $D = \{(t,y) : 1 \le t \le 2, y \in \mathbb{R}\}$.

First, check that $f(t,y) = \frac{1+t}{1+y}$ is jointly continuous. It's the ratio of two continuous functions, so it is continuous everywhere except possibly when the denominator is 0. That would be when $y = -1$. Unfortunately, that is an issue! Therefore, the approach of using Theorem 5.4 doesn't quite work (if students didn't notice this, they still get some partial credit for the solution).

We'll instead use the variant Theorem 1 listed at the end of this PDF. We'll setup $\mathcal{D}$ as in that theorem, using $a = 1$, $T = 1$, $y_0 = 2$ and $R = 1$, since with $R = 1$ we have $y \ge 1$ on this region, so we've excluded the point where f is not continuous, and then

$$M = \max_{(t,y)\in\mathcal{D}} \left|\frac{1+t}{1+y}\right| \le \frac{\max_{0\le t\le 2}|1+t|}{\min_{1\le y\le 3}|1+y|} = \frac{3}{2}$$

and then (assuming we're Lipschitz continuous), applying the theorem, we know we have a solution for $t \in [a, a+\delta]$ where $\delta = \min(T, R/M) = \min(1, 1/\frac{3}{2}) = \frac{2}{3}$, meaning we showed existence and uniqueness for $t \in [1, 1\frac{2}{3}]$. By playing around with making a smaller R we get a smaller M, but the smallest we can make R/M is $2.25/3 < 1$ (using $R = 1.5$) so we can't reach $t = 2$ this way.

Are we Lipschitz continuous? For Theorem 1 we don't even need to know the constant. Then it's sufficient to check that $\partial/\partial y f$ is continuous on $\mathcal{D}$, since then (again by Bolzano-Weierstrass) it follows $|\partial/\partial y f|$ is bounded on $\mathcal{D}$, hence that f is Lipschitz continuous in y uniformly in t. If you wanted, you could compute an explicit bound, since $|\partial/\partial y f(t,y)| = \left|\frac{1+t}{(1+y)^2}\right|$ and $1+t \le 3$ and $(1+y)^2 \ge 1$ so the Lipschitz constant is no bigger than 3.

For the extra credit, how do we reach $t = 2$? We do the problem in two parts, first showing existence and uniqueness on $t \in [1, 1\frac{2}{3}]$, and then we'll show existence and uniqueness

[1] which is *the* standard ODE existence theorem seen in an undergraduate course; it's related to, but not quite the same, as the Picard–Lindelöf theorem aka Cauchy–Lipschitz theorem

for $t \in [1\frac{1}{2}, 2]$, and since these intervals overlap, by uniqueness, we really have one unique solution on $t \in [1, 2]$. How do we do this? We have existence and uniqueness on $t \in [1, 1\frac{2}{3}]$ already, and furthermore, we know $|y - y_0| \leq R$ on this interval, meaning the solution is bounded in $1 \leq y \leq 3$. Therefore $f(t, y) \geq 0$, hence $y' \geq 0$. But if $y(1) = 2$ and $y' \geq 0$, then by the fundamental theorem of calculus, it follows $y(t) \geq 2$ for $t \in [1, 1\frac{2}{3}]$. We need to do this so that we know something about our new "y_0" (which is the value of y a the new "$a = 1\frac{1}{2}$"). We may not know the exact value of $y(1\frac{1}{2})$ but we know it is greater than 2. Thus we can essentially repeat the process we did when we had $a = 1$, and for $a = 1\frac{1}{2}$ we again use $R = 1$ and can bound $M \leq \frac{3}{2}$ so we have existence of a solution for the interval $[1\frac{1}{2}, 1\frac{1}{2} + \frac{2}{3}]$.

That proof was a bit quick and vague. We won't be asking questions like that on the exam, so don't worry!

b) Find a formula for the true solution

Solution:

The right-hand-side is separable, so we can do separation of variables, and solve

$$\int (1 + y)\, dy = \int (1 + t)\, dt$$

which gives

$$\frac{1}{2}y^2 + y = \frac{1}{2}t^2 + t + C$$

and we can multiply this by 2 and then plug in $y(1) = 2$ to find $C = 5$ and we have

$$y^2 + 2y + c = 0, \quad c = -t^2 - 2t - 5.$$

Solve via the quadratic formula to give

$$y = \frac{-2 \pm \sqrt{4 - 4c}}{2} = -1 \pm \sqrt{t^2 + 2t + 6}$$

and we deduce that the solution must be

$$\boxed{y(t) = -1 + \sqrt{t^2 + 2t + 6}}$$

because we can exclude the $-\sqrt{\ldots}$ case since it doesn't satisfy the initial conditions.

c) Apply Euler's method with $h = 0.5$ to estimate $y(2)$. What is the estimate that Euler's method gives? You can include your code if you want (it might help with partial credit) but it's not necessary.

Solution:

It is $\boxed{2.7083}$.

d) Compute the error bound from Theorem 5.9 in Burden and Faires, and compare the bound at $t = 2$ with the actual error at $t = 2$. If some of the constants in Theorem 5.9 are difficult to calculate, you are welcome to use a reasonable bound of them.

Solution:

We find $y(2) - w_2 = 2.7417 - 2.7083 = 0.0333$ so the actual error is 0.0333.

Theorem 5.9 bounds the error at any $t_i = a + ih$ by

$$|y(t_i) - w_i| \leq \frac{hM}{2L}\left[e^{L(t_i - a)} - 1\right]$$

where $y''(t) \leq M$ on $t \in [a, b]$ (not the same "M" as in Theorem 1) and L is the Lipschitz constant.

To find the Lipschitz constant L, we can do a crude approximation, following the reasoning in (a) that y is bounded away from -1 so $y' \geq 0$ and hence $y(t) \geq 2$, so $L = \max |\partial/\partial y f(t,y)| = \max \left| \frac{1+t}{(1+y)^2} \right| \leq \max \frac{3}{3^2} = \frac{1}{3}$, so $L = \frac{1}{3}$ is a valid answer.

We could try a better approximation, using the true value of y. Then $L = \max |\partial/\partial y f(t,y)| = \max \left| \frac{1+t}{(1+y)^2} \right| = \max \left| \frac{1+t}{(1-1+\sqrt{t^2+2t+6})^2} \right| = \max \left| \frac{1+t}{t^2+2t+6} \right|$. From here, we can either do a crude bound, bounding $1+t \leq 3$ and $\frac{1}{t^2+2t+6} \geq 9$ (since $1 \leq t \leq 2$) hence $L \leq \frac{1}{3}$ as we found before. Or, we can actually minimize this, which gives about 0.223, just slightly better (I minimized this via Mathematica).

To find M, we must compute y''. We have $y'(t) = \frac{1}{2}(t^2+2t+6)^{-1/2}(2t+2)$ via the chain rule, and then

$$y''(t) = -\frac{1}{4}\frac{(2t+2)^2}{(t^2+2t+6)^{3/2}} + \frac{1}{(t^2+2t+6)^{1/2}} = \frac{5}{(t^2+2t+6)^{3/2}}$$

doing the product and chain rules, and then simplifying when making a common denominator. This is maximized when the denominator is minimized, and since all the coefficients in the denominator are positive, that means $(t^2+2t+6)^{3/2}$ is increasing, so it's minimized at the smallest t value, so $t = 1$. Thus $M = \frac{5}{(1+2+6)^{3/2}} = \frac{5}{9^{3/2}} = \frac{5}{27}$.

Thus our bound, for $t = 2$, is

$$|y(t_2) - w_2| \leq \frac{hM}{2L}\left[e^{L(2-1)} - 1\right] \leq \frac{.5\frac{5}{27}}{\frac{2}{3}}\left[e^{\frac{1}{3}(2-1)} - 1\right] = \frac{5}{36}\left[e^{1/3} - 1\right] \approx \boxed{0.0549}.$$

So our actual error 0.0333 is less than the bound 0.0549, as it ought to be. If you use the tighter bound of L, we get an error bound of 0.0519, just slightly better.

Note: A big mistake is to solve for L and M in terms of t but not maximize over $t \in [1,2]$ separately for each L and M. If you write them in terms of t and then maximize M/L, that is not correct. We need L and M to be numbers, not functions of t.

e) Now consider $t \in [1, 1+10^{-5}]$. Use Euler's method to estimate $y(1+10^{-5})$ using a variety of stepsizes h (between 10^{-4} and 10^{-14}). Plot the error as a function of stepsize h; choose the parameters of the plot to make it informative. Which choice of h minimizes the error, and is this expected?

Solution:

See Fig. 3(left). Your plot might not look exactly the same, but it should show a minimum in error, and the minimum should be between $h \in [10^{-11}, 10^{-7}]$ and probably within the range $h \in [10^{-10}, 10^{-8}]$. This is expected, because for very small h, roundoff error shows up, and we don't just look at Theorem 5.9, we look at Theorem 5.10 since this incorporates roundoff error. We have a δ/h term, so we can see that for very small h, this δ/h term starts to dominate. This is much like we saw for finite difference formulas.

f) Repeat the same exercise as above, but for $t \in [1,2]$. Again, choose a variety of stepsizes, but don't make h so small that it takes more than a few seconds for your code to run. Plot the error. Which choice of h minimizes the error? Are there any differences with your result compared to the previous question?

Solution:

See Fig. 3(right). We only go to about 10^{-10} stepsize (or larger), since this takes about 27 seconds to run on my laptop. Smaller stepsizes just take too long to run.

Now we do *not* see a "sweet spot" for the stepsize: it's just smaller h is better. There really is an optimal stepsize, but it's just that it's so small, that we never see it because it'd be too expensive to run the ODE solver for such a small stepsize on such a (relatively) long time interval.

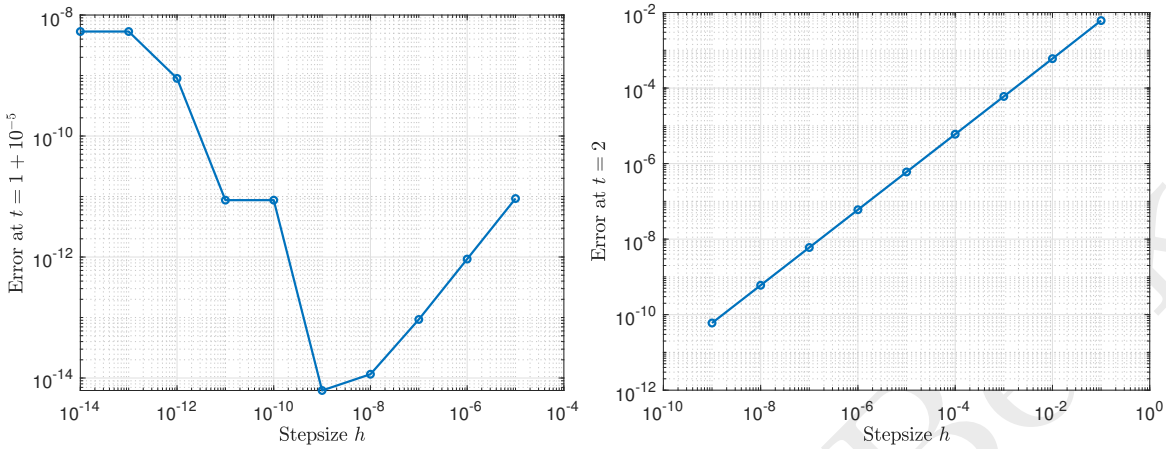

Figure 3: Left: for part 2(e). Right: for part 2(f).

So the same phenomenon as in (e) exists but we don't observe it, because for a practical solver, we cannot choose h to be so small. In this sense, it is not like finite difference methods, since for finite difference methods, it is not more or less costly to choose a small h. For IVP, choosing a small h is much more costly, since we're solving over an entire interval. So we don't run into roundoff errors quite as easily.

Extra ODE existence theorem

There are many quite similar ODE existence theorems, such as Picard–Lindelöf theorem aka Cauchy–Lipschitz theorem. The naming conventions are not quite universal, and the slight differences in assumptions are important. The following theorem (adapted from Thm. 3.10 in Hunter and Nachtergaele's applied analysis) may be useful for us:

Theorem 1 (Local existence for ODEs) *Consider the IVP*

$$y' = f(t, y) \quad \textit{and} \quad y(a) = y_0,$$

and define

$$\mathcal{D} = \{(t, y) \in \mathbb{R}^2 \mid a - T \le t \le a + T,\ y_0 - R \le y \le y_0 + R\}$$

for some $T > 0$. Let $f : \mathcal{D} \to \mathbb{R}$ be continuous; then since D is a closed set, $|f|$ is bounded over $\mathcal{D}$ (this follows from the Bolzano-Weierstrass theorem, an extension of the Extreme Value Theorem to higher dimension), and call this bound M, i.e., $\forall (t, y) \in \mathcal{D}$, $|f(t, y)| \le M$. Further suppose that on $\mathcal{D}$, f is Lipschitz continuous with respect to y uniformly in t. Then there is a unique solution y to the ODE within the time interval $[a, a + \delta]$ where $\delta = \min(T, R/M)$, and furthermore $|y(t) - y_0| \le R$ for all $t \in [a, a + \delta]$.

Problem 1 complete code

Here is the complete code

```
% == Geometry of equilateral triangle
a = 10*sqrt(3)/3;
R1 = [-10;-a]; % healty raptor's initial position
R2 = [10;-a];  % healty raptor's initial position
R3 = [0;2*a];  % injured raptor's initial position

h0  = [0;0]; % Human's initial posiition

```

```
% == Plot the initial geometry
figure(1); clf;
h1=scatter( R1(1), R1(2) , 100, 'o', 'filled','DisplayName','Healthy Raptor' );
hold all
h2=scatter( R2(1), R2(2) , 100, 'o', 'filled','DisplayName','Healthy Raptor' );
h3=scatter( R3(1), R3(2) , 100, 'o', 'filled','DisplayName','Injured Raptor' );
h4=scatter( h0(1), h0(2) , 100, '^', 'filled','DisplayName','Human' );
axis equal
line( [R1(1),R2(1)], [R1(2),R2(2)],'linestyle','--','color','k' )
line( [R3(1),R2(1)], [R3(2),R2(2)],'linestyle','--','color','k' )
line( [R3(1),R1(1)], [R3(2),R1(2)],'linestyle','--','color','k' )
ylim([-6.5,12]);

% == the top-speeds
v_h = 6;
v1  = 25;
v2  = 25;
v3  = 20;

% == the human's path
% angle   = 45; % for example
angle   = 56;
theta = deg2rad(angle);
c = [cos(theta);sin(theta)]; % already normalized
h   = @(t) v_h*t.*c + h0;

% == form RHS of the raptor ODEs
tspan       = [0,1];
F_raptor1   = @(t,y) v1*(h(t)-y)/norm(h(t)-y);
F_raptor2   = @(t,y) v2*(h(t)-y)/norm(h(t)-y);
F_raptor3   = @(t,y) v3*(h(t)-y)/norm(h(t)-y);

% == define a function that stops the ODE solver when the raptor
%    is close enough to the human (within cutoffRadius)
cutoffRadius    = .01; % The raptor catches the human when they're this close
opts = odeset('abstol',1e-4,'event',@(t,y)caught(t,y,h,cutoffRadius) );
% see caught.m

% == solve the ODEs
sol_1       = ode45( F_raptor1, tspan, R1, opts );
sol_2       = ode45( F_raptor2, tspan, R2, opts );
sol_3       = ode45( F_raptor3, tspan, R3, opts );
% == retrieve an interpolant (for plotting)
t_1         = sol_1.x(end);     % when raptor #1 would catch you
t_2         = sol_2.x(end);     % when raptor #2 would catch you
t_3         = sol_3.x(end);     % when raptor #3 would catch you
[t_devour,ind]    = min( [t_1,t_2,t_3] );     % time to first raptor catching you
fprintf('Rapter %d caught you, and it took %.3f seconds\n', ind, t_devour );
raptor1     = @(t) deval( sol_1, t );
raptor2     = @(t) deval( sol_2, t );
raptor3     = @(t) deval( sol_3, t );

% == for plotting
tGrid       = linspace(tspan(1),t_devour);
```

```
r1          = raptor1(tGrid);
r2          = raptor2(tGrid);
r3          = raptor3(tGrid);
human       = h(tGrid);
p1=plot( r1(1,:), r1(2,:), 'o-','color',h1.CData,'linewidth',2,'DisplayName','Healthy raptor
    ','MarkerIndices',[]);
p2=plot( r2(1,:), r2(2,:), 'o-','color',h2.CData,'linewidth',2,'DisplayName','Healthy raptor
    ','MarkerIndices',[]);
p3=plot( r3(1,:), r3(2,:), 'o-','color',h3.CData,'linewidth',2,'DisplayName','Injured raptor
    ','MarkerIndices',[]);
p4=plot( human(1,:), human(2,:), '^-','color',h4.CData,'linewidth',2,'DisplayName','Human','
    MarkerIndices',[]);
grid on
set(gca,'fontsize',20);
legend([p1,p2,p3,p4])
title(sprintf('Human at $%.0f^\\circ$, $t_\\textrm{devour}=%.3f$ s',angle,t_devour),'
    interpreter','latex');

export_fig 'HW8_raptor_56degrees' -pdf -transparent
```

and it relies on this helper function `caught.m`:

```
function [value, isterminal,direction] = caught(t,y,humanTrajectory,cutoff)
% It tells the ODE solver to stop computing once the
% human has been caught by the raptor (this is the event).

% This means that if we have a root of value, then
%   terminate the ODE solver:
isterminal = 1;
% Don't worry about this much:
direction = 0;
% This is the event. We look for a zero-crossing, aka root.
value = norm(humanTrajectory(t) - y ) - cutoff;
```

Here's a similar code in Python:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from scipy.integrate import solve_ivp
mpl.rcParams[figure.figsize] = [7,7]
mpl.rcParams[lines.linewidth] = 2
mpl.rcParams[lines.markersize] = 4
mpl.rcParams.update({'font.size': 20})
mpl.rcParams['mathtext.fontset'] = 'cm'

## Setup the geometry
a = 10*np.sqrt(3)/3
R1 = [-10,-a] # healthy raptor initial position
R2 = [10,-a]  # healthy raptor initial position
R3 = [0,2*a]  # injured raptor initial position
h0 = [0,0]    # Human initial position

# Plot
plt.scatter(R1[0],R1[1],100);
plt.scatter(R2[0],R2[1],100);
```

```
plt.scatter(R3[0],R3[1],100);
plt.scatter(h0[0],h0[1],100);
ax = plt.gca()
ax.axis('equal');
ax.grid()

## Some problem data
v_h = 6  # human's top speed
v_healthy  = 25 # healthy raptor top speed
v_injured  = 20 # injured raptor top speed

# Human's path
angle = 45
theta = np.deg2rad(angle)
c     = np.array( [np.cos(theta),np.sin(theta)] )

def human(t):
  return v_h*t*c + h0

# Raptor's behavior
def F_healthy_raptor(t,y):
  direction = human(t) - y
  return v_healthy*( direction )/np.linalg.norm( direction )

def F_injured_raptor(t,y):
  direction = human(t) - y
  return v_injured*( direction )/np.linalg.norm( direction )

# https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html
#  and see examples at the bottom of that website
cutoffRadius = 0.01
def caught(t,y):
  return np.linalg.norm( human(t) - y ) - cutoffRadius
caught.terminal = True

## Solve the IVP/ODE
tspan    = [0,1]
ivp_soln1= solve_ivp( F_healthy_raptor, tspan, R1, dense_output=True,events=caught,rtol=1e
    -5)
ivp_soln2= solve_ivp( F_healthy_raptor, tspan, R2, dense_output=True,events=caught,rtol=1e
    -5)
ivp_soln3= solve_ivp( F_injured_raptor, tspan, R3, dense_output=True,events=caught,rtol=1e
    -5)
raptor1  = ivp_soln1.sol
raptor2  = ivp_soln2.sol
raptor3  = ivp_soln3.sol
t_1      = ivp_soln1.t_events[0][0]
t_2      = ivp_soln2.t_events[0][0]
t_3      = ivp_soln3.t_events[0][0]
t_devour = np.min( [t_1,t_2,t_3] )
j        = 1+np.argmin( [t_1,t_2,t_3] )
print(f'It takes {t_devour:.3f} seconds for the raptor to catch the human, by raptor {j}')

## Plot the solutions
```

```
t = np.linspace(tspan[0],t_devour,100)

plt.scatter(R1[0],R1[1],100);
plt.scatter(R2[0],R2[1],100);
plt.scatter(R3[0],R3[1],100);
plt.scatter(h0[0],h0[1],100);
ax = plt.gca()
ax.axis('equal');
ax.grid()

r1  = raptor1(t)
r2  = raptor2(t)
r3  = raptor3(t)
h   = np.array([human(ti) for ti in t]).T
plt.plot( r1[0], r1[1], label='Healthy raptor' );
plt.plot( r2[0], r2[1], label='Healthy raptor' );
plt.plot( r3[0], r3[1], label='Injured raptor' );
plt.plot( h[0], h[1], label='Human');
plt.legend(loc='upper right',fontsize=16);
plt.title(f'Human at ${angle}^\circ$, $t={t_devour:.3f}$');
```